AFOSR-TR-97-0061

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
November 23, 1993

3. REPORT TYPE AND DATES COVERED
Final Report: May 1, '94 - Sept. 30, '96

4. TITLE AND SUBTITLE
Fault-tolerant and Real-time Distributed Computing

5. FUNDING NUMBERS
(Grant)
F49620-94-1-0198

6. AUTHOR(S)
Fred B. Schneider

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Cornell University
Ithaca, NY 14853

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR
110 Duncan Ave., Suite B115
Bolling AFB
Washington, DC 20332-0001

NM

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
(Grant)
F49620-94-1-0198

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release;
Distribution Unlimited

12b. DISTRIBUTION CODE
UL

13. ABSTRACT (Maximum 200 words)

Progress was made on a number of problems in the areas of fault-tolerant and real-time computing. Programming logics were investigated for reasoning about distributed programs that must satisfy real-time constraints, must interact with a continuous physical environment, and whose correctness depends on properties of schedulers and degree of resource contention. A new approach to fault tolerance, based on a virtual machine monitor was developed. It provides fault-tolerance without requiring modifications to hardware or software. Finally, software to support mobile network agents was developed and released. Algorithms to implement agent fault-tolerance were developed.

DTIC QUALITY INSPECTED 2

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
SAR

Final Report

Research on Fault-tolerant and Real-time Computing

AFOSR Grant F49620-94-1-0198

Fred B. Schneider

Department of Computer Science
Cornell University
Ithaca, New York 14853

1. Introduction

Under the auspices of this AFOSR funding, research was performed on a variety of topics related to the implementation of fault-tolerant, real-time, and embedded control software for distributed systems. The research had theoretical as well as practical components:

- *Programming logics, formal methods, and programming methodology.* We investigateed programming logics to support the analysis of distributed programs that must satisfy real-time constraints, must interact with a continuous physical environment, and whose correctness depends on properties of schedulers and degree of resource contention.
- *Fault-tolerance.* We invented and prototyped a new approach to supporting replication management. It is based on modifying a virtual machine monitor that runs below an operating system. We have also devised a new approach to analyzing a system's fault-tolerance.
- *Agent-based Computing.* We started studying a new paradigm for structuring distributed systems: the use of mobile agents. A series of prototype systems were implemented and released, and algorithms for constructing fault-tolerant agents were developed.

The work led to 20 publication, which are listed at the end of this report. One patent was granted, and a second patent disclosure was filed and remains pending.

2. Formal Methods

Our work in formal methods was driven by the desire to make logics a usable tool for system developers. Mastery of ordinary first-order logic is required to use most formal methods. Because it is a stumbling block for so many, we investigated ways to make that logic more accessible. This led to our equational presentation of the logic, called E, and a sophomore-level discrete mathematics text. We have since thoroughly explored the axiomatization, use, and teaching of E, and we are slowly generalizing the logic and approach. For example, we developed an equational reasoning apparatus for Dijkstra's "everywhere" (i.e. "is valid") operator. Not only does the new axiomatization extend calculational reasoning to a broader setting, but it reveals a surprising property of logical systems: axiomatizations based on schemas need not be equivalent to employing a substitution inference rule in concert with a finite number of axioms.

The leverage of formal methods is greatest for programming problems that are small and intricate, because brute-force methods do not work there. Process control programs are an example of such programs. For that reason, we investigated methods for extending extant formal methods to this setting. And we discovered two principles for analyzing programs whose executions are affected by an environment. These principles have now been used for verifying real-time behavior of concurrent

programs constrained by schedulers and limited resources; they have also been used for verifying hybrid systems, systems involving continuous physical processes as well as discrete control.

Finally, we made progress on a new characterization of system refinement. The construction of a system usually involves a sequence of steps where abstractions are replaced by implementations. Of interest is a method to establish that this replacement preserves a specification. A variety of methods have been proposed. One, based on refinement mappings, is methodologically attractive. However, it was restricted to specifications that exhibit "finite non determinism" and were formulated as separate safety and liveness properties. We have shown that neither restriction was necessary; we have extended the method and eliminated both requirements.

3. Fault-Tolerance

All schemes for implementing fault tolerance involve some form of replication, where replicas are assumed to fail independently. The key engineering issue that the designer of a fault-tolerant computing system must address is deciding where in the system to implement replica coordination. We have developed a new set of replica-coordination protocols that run below an operating system but above the hardware. With these protocols, a processor can be made fault-tolerant without modifying the hardware, operating system, or application programs. A prototype implementation of the protocols established that their cost was reasonable.

Most reasoning about system fault-tolerance is ad hoc and informal. With large, critical-infrastructure systems, however, this approach and the confidence we can have in its conclusions are unsatisfactory. Therefore, we investigated a new verification framework that is specialized to fault-tolerance. Our framework permits more-natural specifications of fault-tolerance requirements than general-purpose formalisms (e.g. temporal logic). Because it is specialized, the framework supports efficient and mechanized analysis of system fault-tolerance. We implemented an initial prototype software tool based on the framework. The tool has a graphical front end and hides from its users the analysis process itself, making it something that could be included in the "survivable systems toolkit" a system designer might turn to.

4. Operating System Support for Agents

We investigated a new paradigm for structuring distributed systems: mobile agents. The effort involved building operating system support for agents as well as attacking more fundamental problems.

On the practical side, our TACOMA (Tromsoe and Cornell Moving Agents) system is now in daily use as a production platform and runs under HP-UX, Solaris, BSD Unix, and Linux. In contrast to other agent-based approaches, TACOMA supports agents written in a variety of languages. Currently, these languages include C, Java, Perl, Scheme, Python, and Tcl/Tk. A new version of TACOMA, based on HTTP for communications and an ML server, is now being programmed. This HTTP/ML version will make TACOMA a part of the world-wide web and, therefore, broadly accessible.

Our more fundamental agent-based work is driven by the agent-integrity and host-integrity problems. The agent-integrity problem concerns ensuring that an agent computation is successfully completed despite the presence of malicious and faulty hosts. Our work on this problem has led to the study of cryptographic abstractions that can provide fault-tolerance. The host-integrity problem concerns securing hosts so that they cannot be compromised by faulty or hostile agents. Here, we have pursued the use of wrappers and compile-time analysis of agents.

Publications

(1) N. Klarlund and Fred B. Schneider. Proving nondeterministically specified safety properties using progress measures. *Information and Computation 107*,3 (November 1993), 151-170.

(2) Fred B. Schneider. Avoiding AAS Mistakes. *Proceedings of the Air Traffic Management Workshop* (eds. L. Tobais, M. Tashker, A. Boyle). NASA Conference Publication 10151, NASA Ames Research Center, 133-149.

(3) Keith Marzullo, Fred B. Schneider, Jon Dehn. Refinement for fault-tolerance: An aircraft hand-off protocol. *Foundations of Ultradependable Parallel and Distributed Computing, Paradigms for Dependable Applications*, Kluwer Academic Publishers, 1994, 39-54.

(4) L. Fix and Fred B. Schneider. Reasoning about Programs by exploiting the environment. *Proc. 21st International Colloquium, ICALP'94* (Jerusalem, Israel, July 1994), Lecture Notes in Computer Science, Volume 820, Springer-Verlag, New York, 328-339.

(5) L. Fix and Fred B. Schneider. Hybrid verification by exploiting the environment. *Formal Techniques in Real Time and Fault Tolerant Systems* (Luebeck, Germany, September 1994), Lecture Notes in Computer Science, Volume 863, Springer-Verlag, New York, 1-18.

(6) Fred B. Schneider. A role for formal methodists. *Dependable Computing and Fault-Tolerant Systems* Vol. 9 (eds. F. Cristian, G. LeLann, T. Lunt), Springer-Verlag, 1995, 43-45.

(7) Scott Stoller and Fred B. Schneider. Verifying programs that use causally-ordered message-passing. *Science of Computer Programming 24*,2 (1995), 105-128.

(8) Fred B. Schneider. On Traditions in Marktoberdorf. *Deductive Program Design.* (M. Broy, ed.) ASI Vol. F152. Springer-Verlag, Heidelberg, 1-4.

(9) Fred B. Schneider. Notes on Proof Outline Logic. *Deductive Program Design.* (M. Broy, ed.) ASI Vol. F152. Springer-Verlag, Heidelberg, 351-394

(10) David Gries and Fred B. Schneider. Avoiding the undefined by underspecification. *Computer Science Today Recent Trends and Developments* (Jan van Leeuwen, ed). Lecture Notes in Computer Science, Vol. 1000, Spring-Verlag, 1995, 366-373.

(11) D. Gries and Fred B. Schneider. Equational propositional logic. *Information Processing Letters 53*,3 (February 1995), 145-152.

(12) Dag Johansen, Robbert van Renesse, and Fred B. Schneider. Operating system support for mobile agents. *Proc. Fifth Workshop on Hot Topics in Operating Systems HOTOS-V* (Orcas Island, Washington, May 1995), 42-45.

(13) David Gries and Fred B. Schneider. A new approach to discrete teaching mathematics. *Primus V*,2 (June 1995), 113-138.

(14) Scott Stoller and Fred B. Schneider. Faster possibility detection by combining two approaches. *Proc. 9th International Workshop, WDAG '95* (Le Mont-Saint-Michel, France, Sept. 1995), Lecture Notes in Computer Science, Volume 972, Springer-Verlag, New York, 1995, 318-332.

(15) David Gries and Fred B. Schneider. Teaching math more effectively, through the design of calculational proofs. *The Mathematical Monthly* (October 1995), 691-697.

(16) Thomas C. Bressoud and Fred B. Schneider. Hypervisor-based Fault Tolerance. *Proc. Fifteenth ACM Symposium on Operating Systems Principles* (Copper Mountain Resort, Colorado, Dec. 1995), Operating Systems Review Vol. 29, No. 5, 1-11.

(17) Thomas C. Bressoud and Fred B. Schneider. Hypervisor-based Fault Tolerance. *ACM Transactions on Computer Systems 14*, 1 (Feb. 1996), 80-107.

(18) Yaron Minsky, Robbert van Renesse, Fred B. Schneider and Scott Stoller. *Proc. of the Seventh ACM SIGOPS European Workshop "System Support for Worldwide Applications"* (Connemara, Ireland, Sept. 1996), ACM, New York. 109-114.

(19) Dag Johansen, Robbert van Renesse, and Fred B. Schneider Supporting broad internet access to TACOMA. *Proc. of the Seventh ACM SIGOPS European Workshop "System Support for*

Worldwide Applications" (Connemara, Ireland, Sept. 1996), ACM, New York. 55-58.

(20) David Gries and Fred B. Schneider Adding the Everywhere Operator to Propositional Logic. Submitted to *The Journal of Logic and Computation.*

Patents

(1) Fault tolerant computer system with shadow virtual processor. United States patent number 5,488,716, Jan. 30, 1996. Co-inventors: E. Balkovich, B. Lampson, and D. Thiel.

(2) Transparent fault tolerant computer system. Patent disclosure filed Dec 1, 1995. Co-inventors: Thomas C. Bressoud, John E. Ahern, Kenneth P. Birman, Robert C.B. Cooper, Bradford B. Glade, and John D. Service.